USB Bus Interface Chip CH375

Datasheet (II): USB Basic Transmission Commands

Version: 4

http://wch.cn

1. Additional Commands

Code	Command Name	Input Data	Output Data	Command Purpose
04H	SET_USB_SPEED	Bus speed		Set USB bus speed
0AH	GET_DEV_RATE	Data 07H	Data rate	Get the data rate type of the USB device
0BH	SET_RETRY	Data 25H Number of retries		Set the number of retries for USB transaction operations
0FH	DELAY_100US		Delay status	Delay 100uS
13H	SET_USB_ADDR	Address value		Set USB address
1CH	SET_ENDP6	working mode	(Wait for 3uS)	Set the receiver for USB host endpoint
1DH	SET_ENDP7	working mode	(Wait for 3uS)	Set the transmitter for USB host endpoint
27H	RD_USB_DATA0		Data Length Data stream	Read the data block from the endpoint buffer of the current USB interrupt
41H	CLR_STALL	Endpoint number	Generate interrupt	Control transmission: Clear endpoint error
45H	SET_ADDRESS	Address value	Generate interrupt	Control transmission: Set USB address
46H	GET_DESCR	Descriptor type	Generate interrupt	Control transmission: Get the descriptor
49H	SET_CONFIG	Configuration Value	Generate interrupt	Control transmission: Set USB configuration
4DH	AUTO_SETUP		Generate interrupt	Automatically configure USB device
4EH	ISSUE_TKN_X	Sync flag Transaction attribute	Generate interrupt	Send synchronization token, and execute transactions
4FH	ISSUE_TOKEN	Transaction attribute	Generate interrupt	Send token, and execute transactions
50H	DISK_BOC_CMD		Generate interrupt	Execute BulkOnly transport protocol commands
52H	DISK_RESET		Generate interrupt	Reset USB storage device
5DH	DISK_MAX_LUN		Generate interrupt	Get the maximum unit number of the USB storage device

If the input data is the working mode of transceiver for USB endpoint, refer to the following table.

Working mode byte	Name	Bit analysis description of working mode	
Bits 7-6	Synchronous trigger flag	If the bit 7 is 1, the bit 6 will be the new synchronous trigger flag:	
		00 or 01 = Keep the current synchronous trigger flag unchanged	
		10 = Synchronous trigger flag set to 0	11 = Synchronous trigger flag set to 1
Bits 5-4	(Reserved bit)	(Undefined, must be 0)	
Bits 3-0	Transaction response mode	Must be 0000	

1.1. Command SET_USB_SPEED

This command is used to set the USB bus speed (this function is not supported on some models of chips). This command needs the input of 1 data to select the USB bus speed. 00H corresponds to 12Mbps full speed mode, and 02H corresponds to 1.5Mbps low speed mode. The USB bus speed of CH375 is 12Mbps full speed by default, and will be restored to 12Mbps full speed mode after the command SET_USB_MODE is executed to set USB working mode.

1.2. Command GET_DEV_RATE

This command is used to get the data rate type of the currently connected USB device. This command requires to input one data 07H, output the data rate type. If the bit 4 is 1, the device will be 1.5Mbps low speed USB device; otherwise, it will be12Mbps full speed USB device. This command is only valid in USB mode 5 (enabled USB host mode, not generating SOF package).

1.3. Command SET_RETRY

This command is used to set the number of retries for USB transaction operations. This command requires to input two data, respectively the data 25H and the number of retries.

The bits 7 and 6 of the number of retries specify the processing mode when CH375 receives NAK response. If the bit 7 is 1 and the bit 6 is 0, infinite retry will be performed (the current retry can be given up through the command ABORT_NAK); if the bit 7 is 1 and the bit 6 is 1, finite retry will be performed for 200mS to 2S around; if the bit 7 is 0, NAK will be notified to MCU as the result or processed as an error. Bits 5-0 of the number of retries specify the number of retries CH375 after the USB device response timeout. If the bit is 0, retry will not be performed after timeout.

The default number of retries is 85H after chip reset or USB mode reset, so infinite retry is performed after NAK reply, and the response is received, and 5 retries will be performed after USB device response timeout.

1.4. Command DELAY_100US

This command is used to delay for 100uS and only supports parallel port mode. The parallel port output data is 0 during delay, and is non-0 (usually the chip version number) at the end of the delay. MCU determines whether the delay is ended according to the read data.

1.5. Command SET_USB_ADDR

This command is used to set the USB device address. This command requires to input 1 data to select the address of the USB device being operated. After reset or after the USB device is connected or disconnected, the USB device address is always 00H, and MCU communicates with the USB device through the default address 00H. If MCU sets the address of the USB device through the standard USB request, it must set the

same USB device address through the command, so that CH375 can communicate with the USB device through the new address.

1.6. Command SET_ENDP6

This command is used to set the receiver for USB host endpoint or the endpoint 2. This command needs to input 1 data to specify a new working mode. For example, if an IN transaction is executed, DATA0 is expected to be received and DATA1 is given up, the synchronous trigger flag of the receiver at the host endpoint must be set to 0 through this command. The byte in the corresponding working mode byte is 80H. Typically, this command is completed within 3uS.

1.7. Command SET_ENDP7

This command is used to set the transmitter for USB host endpoint or the endpoint 2. This command needs to input 1 data to specify a new working mode. For example, if SETUP or OUT transaction is executed and DATA0 is expected to be transmitted, the synchronous trigger flag of the transmitter at the host endpoint must be set to 0 through this command. The byte in the corresponding working mode byte is 80H. If DATA1 is expected to be transmitted, the working mode byte will be C0H. Typically, this command is completed within 3uS.

1.8. Command RD_USB_DATA0

This command is used to read the data block from the endpoint buffer of the current USB interrupt. In USB host mode, this command is exactly the same as the command RD_USB_DATA in function, except that this command is slightly more efficient.

1.9. Command CLR_STALL

This command is used to clear the wrong control transmission command of the endpoint. This command requires to input 1 data to specify the endpoint address of the USB device for which errors will be cleared. Valid addresses are 01H ~ 0FH for OUT endpoint and 81H ~ 8FH for IN endpoint. This command is used to simplify the standard USB request CLEAR_FEATURE. CH375 requests an interrupt from MCU after the command is executed. MCU can read the interrupt status as the operation status of this command. If the operation status is USB_INT_SUCCESS, the command will be executed successfully. Otherwise, the command will be executed unsuccessfully.

1.10. Command SET_ADDRESS

This command is used to set the control transmission command of the USB address. This command requires to input 1 data to specify the new address USB device, and the valid addresses are 00H-7FH. This command is used to simplify the standard USB request SET_ADDRESS. CH375 requests an interrupt from MCU after the command is executed. MCU can read the interrupt status as the operation status of this command. If the operation status is USB_INT_SUCCESS, the command will be executed successfully. Otherwise, the command will be executed unsuccessfully.

1.11. Command GET_DESCR

This command is used to get the wrong control transmission command of the descriptor. This command requires to input 1 data to specify the type of descriptor to be obtained. The valid type is 1 or 2, corresponding to DEVICE descriptor and CONFIGURATION descriptor. CONFIGURATION descriptor also includes the interface descriptor and the endpoint descriptor. This command is used to simplify the standard USB request GET_DESCRIPTOR. CH375 requests an interrupt from MCU after the command is executed. MCU can read the interrupt status as the operation status of this command. If the operation status is USB_INT_SUCCESS, the command will be executed successfully. Otherwise, the command will be executed unsuccessfully. As the control transmission buffer of CH375 is only 64 bytes, CH375will return the operation status USB_INT_BUF_OVER when the length of the descriptor exceeds 64 bytes. For this USB

device, MCU can process the control transmission through the command ISSUE_TOKEN or ISSUE_TKN_X.

1.12. Command SET_CONFIG

This command is used to set the control transmission command of the USB configuration. This command requires to input 1 data to specify a new USB configuration value. If the configuration value is 0, the configuration will be canceled, otherwise it shall be taken from the configuration descriptor of the USB device. This command is used to simplify the standard USB request SET_CONFIGURATION. CH375 requests an interrupt from MCU after the command is executed. MCU can read the interrupt status as the operation status of this command. If the operation status is USB_INT_SUCCESS, the command will be executed successfully. Otherwise, the command will be executed unsuccessfully.

1.13. Command AUTO_SETUP

This command is used to automatically configure USB device. This command is used to simplify the initialization steps for the common USB device and is equivalent to the sequence of multiple commands such as GET_DESCR, SET_ADDRESS, SET_CONFIGURATION and so on. CH375 requests an interrupt from MCU after the command is executed. MCU can read the interrupt status as the operation status of the command. If the operation status is USB_INT_SUCCESS, the command will be executed successfully. Otherwise, the command will be executed unsuccessfully.

1.14. Command ISSUE_TKN_X

This command enables CH375 to send a synchronous token to execute the transaction. This command requires to input two data, respectively synchronization flag and transaction attribute. The bit 7 of the synchronization flag is the synchronous trigger flag of the receiver at the host endpoint, the bit 6 is the synchronous trigger flag of the transmitter at the host endpoint, and the bits 5-0 must be 0. The low 4 bits of the transaction attribute specify the token PID of the transaction, and the high 4 bits specify the destination endpoint number of the USB device. CH375 requests an interrupt from MCU after the command is executed. MCU can read the interrupt status as the operation status of the command. If the operation status is USB_INT_SUCCESS, the command will be executed successfully. Otherwise, the command will be executed unsuccessfully. The only difference between this command and the command ISSUE_TOKEN is that this command always sets the synchronous trigger flag before executing a transaction (equivalent to adding the command SET_ENDP?).

1.15. Command ISSUE_TOKEN

This command enables CH375 to send a token to execute the transaction. This command requires to input 1 data as the transaction attribute. The low 4 bits of the transaction attribute specify the token PID of the transaction, and the high 4 bits specify the destination endpoint number of the USB device. CH375 requests an interrupt from MCU after the command is executed. MCU can read the interrupt status as the operation status of the command. If the operation status is USB_INT_SUCCESS, the command will be executed successfully; otherwise, the command will be executed unsuccessfully. MCU can further analyze the cause of failure according to the operation status.

For SETUP and OUT transactions sending data, first write the data to be sent through the command WR_USB_DATA7, and then execute the transaction through the command ISSUE_TOKEN. For IN transaction receiving data, first execute the transaction through the command ISSUE_TOKEN and read the received data through the command RD_USB_DATA after successful execution of transaction.

For example, when the transaction attribute byte is 09H, CH375 receives data from the default endpoint 0 of the USB device; when the transaction attribute byte is 21H, CH375 sends data to the endpoint 2 of the USB device; when the transaction attribute byte is 29H, CH375 receives data from the endpoint 2 of the USB device, with address of 82H.

The following table shows USB token PID supported by CH375.

PID byte	Name	Description
0DH	DEF_USB_PID_SETUP	Initiate control transmission, and send setup data
01H	DEF_USB_PID_OUT	Execute OUT transaction, and send data
09H	DEF_USB_PID_IN	Execute IN transaction, and receive data

1.16. Command DISK_BOC_CMD

This command is used to execute BulkOnly transport protocol commands for USB storage devices. Before executing this command, MCU must first write the corresponding CBW package to CH375 through the command WR_USB_DATA7. CH375 requests an interrupt from MCU after the command is executed. MCU can read the interrupt status as the operation status of this command. If the operation status is USB_INT_SUCCESS, the command will be executed successfully. For the operation with returned data, the returned data can be obtained through the command RD_USB_DATA.

1.17. Command DISK_RESET

This command is used to reset USB storage device through the control transmission. CH375 requests an interrupt from MCU after the command is executed. MCU can read the interrupt status as the operation status of the command. If the operation status is USB_INT_SUCCESS, the command will be executed successfully. When an error occurs on the USB storage device, CH375 will analyze the cause of the error and automatically select whether the USB device is reset or not as required.

The complete reset process includes: resetting the USB storage device through this command, resetting Bulk-IN endpoint through the command CLR_STALL, and resetting Bulk-OUT endpoint through the command CLR_STALL.

1.18. Command DISK_MAX_LUN

This command is used to get the maximum logical unit number of the USB storage device through control transmission. CH375 requests an interrupt from MCU after the command is executed. MCU can read the interrupt status as the operation status of the command. If the operation status is USB_INT_SUCCESS, the data can be gotten through the command RD_USB_DATA. The data is usually 1 byte.

2. External Firmware

2.1. Overview

The command ISSUE_TOKEN or ISSUE_TKN_X is used to perform basic USB transmission transactions. It is the most basic operation in firmware programming in the USB host mode.

On this basis, the external MCU can process the control transmission for which CH375 has not directly provided the simplified commands according to the requirements of USB protocol. Further, MCU can process the USB protocol of a variety of specific types of devices in accordance with the requirements of the USB protocol, to achieve the control and data exchange of USB devices. CH375 has Bulk-Only transport protocol for Mass-Storage devices. For USB Storage devices using CBI transport protocol, the external MCU still needs to process based on the command ISSUE_TOKEN or ISSUE_TKN_X and the control transmission command.

2.2. External Firmware Reference Flow

The external firmware reference program is provided in the CH375 evaluation board data. The following process is that the external MCU executes the standard USB request GET_STATUS through the control transmission to get the status of USB device for reference when the external MCU designs the firmware program.

(1) Setup stage of control transmission

① Send the command WR_USB_DATA7 to write 8 bytes of request data to the output buffer. The request data is 80H, 00H, 00H, 00H, 00H, 02H and 00H in order, and the length is 8.

② Send the command ISSUE_TKN_X to execute a transaction, with a synchronous flag of 00H and a transaction attribute byte of 0DH. Send a SETUP token to the default endpoint 0 and send DATA0. If the command ISSUE_TOKEN is used to execute a transaction, first set the synchronous trigger flag of the transmitter at the host endpoint to 0 through the command SET_ENDP7 with working mode byte of 80H.

③ MCU waits for the completion of transaction interrupt or wait for the interrupt notification.

④ After the transaction is completed, CH375 sets INT# pin to low level and requests an interrupt from MCU.

⑤ MCU enters the interrupt service program, or exits after receiving the interrupt notification in the main program.

⑥ Send the command GET_STATUS to get the interrupt status.

⑦ CH375 restores INT# pin to a high level and cancels the interrupt request after the command GET_STATUS is completed.

⑧ MCU analyzes the obtained interrupt status. If it is not USB_INT_SUCCESS, the operation will fail and exception will be handled; if it is USB_INT_SUCCESS, the transaction will be executed successfully and the setup stage will be completed.

(2) Data stage of control transmission

① Send the command ISSUE_TKN_X to execute a transaction with the synchronization flag of 80H and the transaction attribute byte of 09H, send an IN token to the default endpoint 0, and send DATA1.

② MCU waits for the transaction to be completed; after the transaction is completed, CH375 requests an interrupt from MCU.

③ Send the command GET_STATUS to get the interrupt status. CH375 cancels the interrupt request.

④ MCU analyzes the interrupt status. If the operation failed, exception will be handled; if it is USB_INT_SUCCESS, the transaction will be executed successfully.

⑤ Send the command RD_USB_DATA0 to get the data returned by the USB device and save it as the return result of the control transmission.

⑥ Because this control transmission requires only one IN transaction, the data stage is completed.

(3) Status stage of control transmission

① Send the command WR_USB_DATA7 to write the status data of the length 0 to the output buffer. The length is 0.

② Send the command ISSUE_TKN_X to execute a transaction with the synchronization flag of 40H and the transaction attribute byte of 01H, send an OUT token to the default endpoint 0 and send DATA1.

③ MCU waits for the transaction to be completed; after the transaction is completed, CH375 requests an interrupt from MCU.

④ Send the command GET_STATUS to get the interrupt status. CH375 cancels the interrupt request.

⑤ MCU analyzes the obtained interrupt status. If the operation failed, exception will be handled; if it is USB_INT_SUCCESS, the transaction will be executed successfully and the status stage will be completed.

(4) The control transmission is completed. The data returned in the data stage is returned as the standard USB request GET_STATUS. Typically, the length of returned data is 2 bytes.